Environmental Modelling and Software 200 (2026) 106957

Contents lists available at ScienceDirect

Environmental Modelling and Software

journal homepage: www.elsevier.com/locate/envsoft

Portable multi-GPU urban flood modelling via MPI-OpenACC: evaluating acceleration, scaling, and workload-per-GPU

Bomi Kim [a], Hyungon Ryu [b], Seungsoo Lee [c], Jun-Hak Lee [d,e], Seong Jin Noh [a,*]

[a] Civil Engineering Department, Kumoh National Institute of Technology, Gumi-si, Gyeongbuk, 39177, Republic of Korea
[b] NVIDIA Corporation, Seoul, Republic of Korea
[c] Korea Environment Institute, Sejong-si, Republic of Korea
[d] School for Environment and Sustainability, University of Michigan, Ann Arbor, MI, 48109, United States
[e] Ojeong Resilience Institute, Korea University, Seoul, 02841, Republic of Korea

ARTICLE INFO

Keywords:
Urban flood modelling
Multi-GPU acceleration
MPI
OpenACC

ABSTRACT

We present a multi-GPU implementation of the physics-based 2D urban flood model (H12) utilising MPI-OpenACC, demonstrating significant reductions in runtime for high-resolution, city-scale, and large-basin simulations. Applied to grids ranging from 1 to 20 m across two basins (downtown Portland, Oregon, USA; downstream Han River, South Korea), the multi-GPU configuration achieved a speedup of approximately 710 × over a serial baseline and approximately 30 × over a 144 core CPU-hybrid run. This advancement renders metre-scale, city-wide scenarios and extensive domains computationally feasible. Our findings provide practical guidance for determining optimal GPU counts based on workload per device; while exact thresholds vary with problem size (spatial coverage and resolution) and system interconnect, the underlying principle aids in aligning domains with available accelerators. The directive-based MPI-OpenACC design maintains a unified code base across CPU and GPU execution, thereby supporting portability and operational use.

1. Introduction

Effective urban flood management necessitates predictive detail at the street level, as hydrodynamic models must resolve flow pathways at the metre-scale, which are influenced by kerbs, road camber, and building footprints, to identify hazards and facilitate actions such as traffic routing, temporary road closures, and pump operations (Fewtrell et al., 2011; Schubert and Sanders, 2012). Recent research has advanced city-scale, high-resolution urban flood modelling towards operational nowcasting; however, these simulations remain computationally intensive and time-sensitive, thereby necessitating the use of graphics processing units (GPUs) and other high-performance computing (HPC) strategies (Xu et al., 2015; Cea and Costabile, 2022; Ming et al., 2025).

Historically, flood models at both city-scale and basin-scale have been parallelised on CPUs through domain decomposition utilising MPI and, in certain instances, OpenMP. While this methodology is scalable on clusters, it frequently results in sub-linear efficiency due to increasing communication overhead (Noh et al., 2018; Chen et al., 2025). Over the past decade, numerous research groups have transitioned computationally intensive shallow-water kernels to GPUs, reporting significant speedups across dam-break, riverine, coastal, and urban environments. This includes both single- and multi-GPU implementations such as SW2D-GPU, TRITON, and other integrated frameworks (Carlotto et al., 2021; Morales-Hernández et al., 2021; Buttinger-Kreuzhuber et al., 2022; Rautenbach et al., 2022; Zhang et al., 2025). Recent advancements include applications at city-scale and basin-scale, as well as heterogeneous CPU–GPU strategies designed for operational deployment. Furthermore, evidence of multi-GPU parallelisation and scaling on both structured and unstructured meshes continues to accumulate (Morales-Hernández et al., 2021; Xue et al., 2024; Dong et al., 2025a; Zhang et al., 2025).

Two primary methodologies are employed for GPU enablement. CUDA offers low-level control and well-established tools. However, it restricts the code base to a single vendor and often necessitates significant refactoring, which can complicate maintenance across heterogeneous systems (Hou et al., 2020; Carlotto et al., 2021; Xie et al., 2025). In contrast, directive-based OpenACC maintains a single source and facilitates both CPU-only and GPU-accelerated builds through compiler options, thereby enhancing portability and long-term maintainability while still supporting multi-GPU execution via MPI with CUDA-aware

* Corresponding author. Global Hall Rm 302, 61 Daehak Ro, Gumi-si, Gyeongbuk, 39177, Republic of Korea.
E-mail address: seongjin.noh@kumoh.ac.kr (S.J. Noh).

https://doi.org/10.1016/j.envsoft.2026.106957
Received 17 October 2025; Received in revised form 13 February 2026; Accepted 10 March 2026
Available online 10 March 2026

communication (Saleem and Norman, 2024). Beyond flood modelling, directive-based acceleration has gained prominence in computational fluid dynamics and eco-hydraulics, highlighting its broader applicability (Xue et al., 2024; Yang et al., 2025). Similar MPI-OpenACC multi-GPU implementations have also been reported in structured heat-transfer and fluid solvers, where communication-computation overlap and scalability across multiple devices have been evaluated (Xu and Li, 2023, 2024).

Despite these advancements, significant gaps persist in the domain of urban flood modelling. Existing GPU-focused studies frequently highlight results from single-device implementations or predominantly report multi-GPU scaling in non-urban contexts. Within the references compiled here, GPU acceleration is indeed represented in urban applications; however, explicit multi-GPU results for urban inundation are comparatively scarce, and directive-based OpenACC implementations for multi-GPU urban flood modelling are particularly underrepresented. OpenACC efforts tend to focus on general shallow-water flows rather than urban testbeds (Morales Hernández et al., 2021; Buttinger Kreuzhuber et al., 2022; Saleem and Norman, 2024; Zhang et al., 2025). Furthermore, there is a paucity of practical guidance on selecting appropriate GPU device counts across varying city-scale domains and resolutions, which underscores the need to synthesise workload-per-GPU considerations from systematic experiments (Dong et al., 2025b).

This study presents a portable, directive-based, multi-GPU implementation of the H12 2D urban flood model utilising MPI and OpenACC and assesses its performance and scalability across two distinct domains. The framework is applied to a pluvial, metre-resolution urban scenario in downtown Portland, USA, and a large-domain case in the downstream Han River basin, South Korea, where both fluvial and pluvial influences may coexist (Noh et al., 2018; Lee et al., 2025). We quantify runtime scaling in relation to device counts and problem sizes, and provide synthesised guidance on workload-per-GPU for practical configuration.

The remainder of this paper is organised as follows. Section 2 describes the H12 urban flood model and the MPI–OpenACC multi-GPU acceleration framework. Section 3 presents the study areas, datasets, and experimental setup. Section 4 reports simulation results, runtime benchmarking, and scaling analyses across resolutions and device counts. Finally, Section 5 concludes the study and outlines potential directions for future work.

2. Methodology

This section describes the urban flood model employed, the development of a multi-GPU-accelerated hybrid parallel code, and the evaluation metrics for computational benchmark.

2.1. H12 urban flood model

The H12 urban flood model is a fully coupled framework that integrates a 2D surface flow model with a 1D sewer network model. To advance understanding of urban flooding, H12 has been applied and extended to various regions in several previous studies (Lee et al., 2016, 2025; Noh et al., 2018; Choi et al., 2025). Leveraging a hybrid parallel computing strategy that combines MPI and OpenMP, H12 achieves accelerated 2D surface flow simulations, enabling efficient high-resolution modelling (Noh et al., 2018). The GPU-based hybrid parallel code developed in this study targets the 2D surface flow component of H12. Although sewer surcharge and inlet capacity are important drivers of urban inundation, this study focuses on the 2D surface component to isolate GPU performance characteristics. The 2D urban flood model used in this study is governed by equations (1)–(3).

$$\frac{\partial h}{\partial t}+\frac{\partial M}{\partial x}+\frac{\partial N}{\partial y}=r_e-q_{ex} \tag{1}$$

$$\frac{\partial M}{\partial t}+\frac{\partial (uM)}{\partial x}+\frac{\partial (vM)}{\partial y}=-gh\frac{\partial H}{\partial x}-\frac{g\cdot rc_{sur}{}^2 M\sqrt{u^2+v^2}}{h^{4/3}} \tag{2}$$

$$\frac{\partial N}{\partial t}+\frac{\partial (uN)}{\partial x}+\frac{\partial (vN)}{\partial y}=-gh\frac{\partial H}{\partial y}-\frac{g\cdot rc_{sur}{}^2 N\sqrt{u^2+v^2}}{h^{4/3}} \tag{3}$$

h is water depth, H is water level, u,v are x,y directional velocity, $M\,(=uh), N\,(=vh)$ are x,y directional flow flux, r_e is effective rainfall and perturbed by noise. The considered runoff rate is in the range of 0.5 to 1.0. q_{ex} represents the drainage discharge per unit area from the ground surface into the drain channel. In this study, q_{ex} are considered zero because only the 2D surface flow component is targeted as stated earlier. g is gravity acceleration, rc_{sur} is Manning's roughness coefficient for the urban surface. The two-dimensional shallow water equations are computed using a finite difference method (FDM) formulated on a staggered grid system. For temporal discretization, a leap-frog integration scheme is employed to achieve second-order accuracy in time. The advection terms are represented by a first-order upwind approximation, while central differencing is utilised to evaluate pressure gradients. Numerical stability is maintained by adapting the time step according to the Courant-Friedrichs-Lewy (CFL) criterion. To manage wet-dry transitions, cells where the water depth (h) becomes less than 0.0001 m are assigned zero velocity and depth, thereby excluding them from the active computation. A more detailed description of the H12 model can be found in previous studies (Lee et al., 2016, 2025; Noh et al., 2018). Settings for the runoff coefficients, Manning roughness, and CFL control for CPU-GPU parallel benchmark runs are provided in Supplementary S1. The computational domain is represented as a structured rectangular grid for each MPI subdomain and stored as 1D arrays in memory. Irregular basin boundaries are handled using a mask field that distinguishes cells. All flux and continuity updates are performed only on active cells through conditional checks within the computational loops. This structured-grid layout simplifies halo exchange and supports efficient memory access on GPUs.

2.2. Multi-GPU acceleration with MPI and OpenACC

In the development of a GPU-accelerated version of the code, we implemented a design utilising MPI and OpenACC, allowing the same source code to be compiled for either CPU-only or GPU-accelerated execution. The MPI domain decomposition of the 2D surface grid was preserved, with each rank managing a contiguous subdomain and exchanging halo cells with neighboring ranks through non-blocking point-to-point communication. A 1D slab decomposition was adopted rather than a 2D block partitioning. This choice maintains a simple and regular subdomain structure that aligns well with the array-based implementation of the solver and limits each rank to communication with only two neighboring ranks, thereby simplifying halo exchange. Although 2D partitioning can reduce the surface-to-volume ratio at very large process counts, the GPU configurations evaluated in this study did not require the additional communication complexity. In the tested configurations (up to eight GPUs), kernel execution dominated runtime and the simpler two-neighbor communication pattern of the 1D slab decomposition provided a favourable balance between scalability and implementation complexity. The CPU-only configuration employs OpenMP threads within each rank, while the GPU configuration offloads compute-intensive loops to NVIDIA devices via OpenACC, maintaining the MPI layer unchanged. The data layout follows a Structure of Arrays design. Each state variable (e.g., water depth, velocity components, and fluxes) is stored in an independent array. This arrangement is well suited to stencil-based finite-difference updates and supports efficient memory access on GPUs. Halo regions are exchanged explicitly using non-blocking MPI point-to-point communication. Interior stencil updates are executed on the GPU via OpenACC parallel loop directives, while halo synchronisation is completed before boundary cell updates. State

arrays are accessed by device kernels through unified memory. Unified memory was adopted primarily for portability and code maintainability rather than necessity. To enhance GPU efficiency, we introduced two practical features. Firstly, a brief GPU warm-up phase executes inexpensive operations to initialise the device context, thereby mitigating first-step latency. Secondly, a straightforward load allocation step assesses the number of surface cells and GPUs, distributing subdomains to ensure each device receives an approximately equal workload. This balanced distribution improved device utilisation across runs. Fig. 1a illustrates the surface domain divided into slabs assigned to MPI ranks,

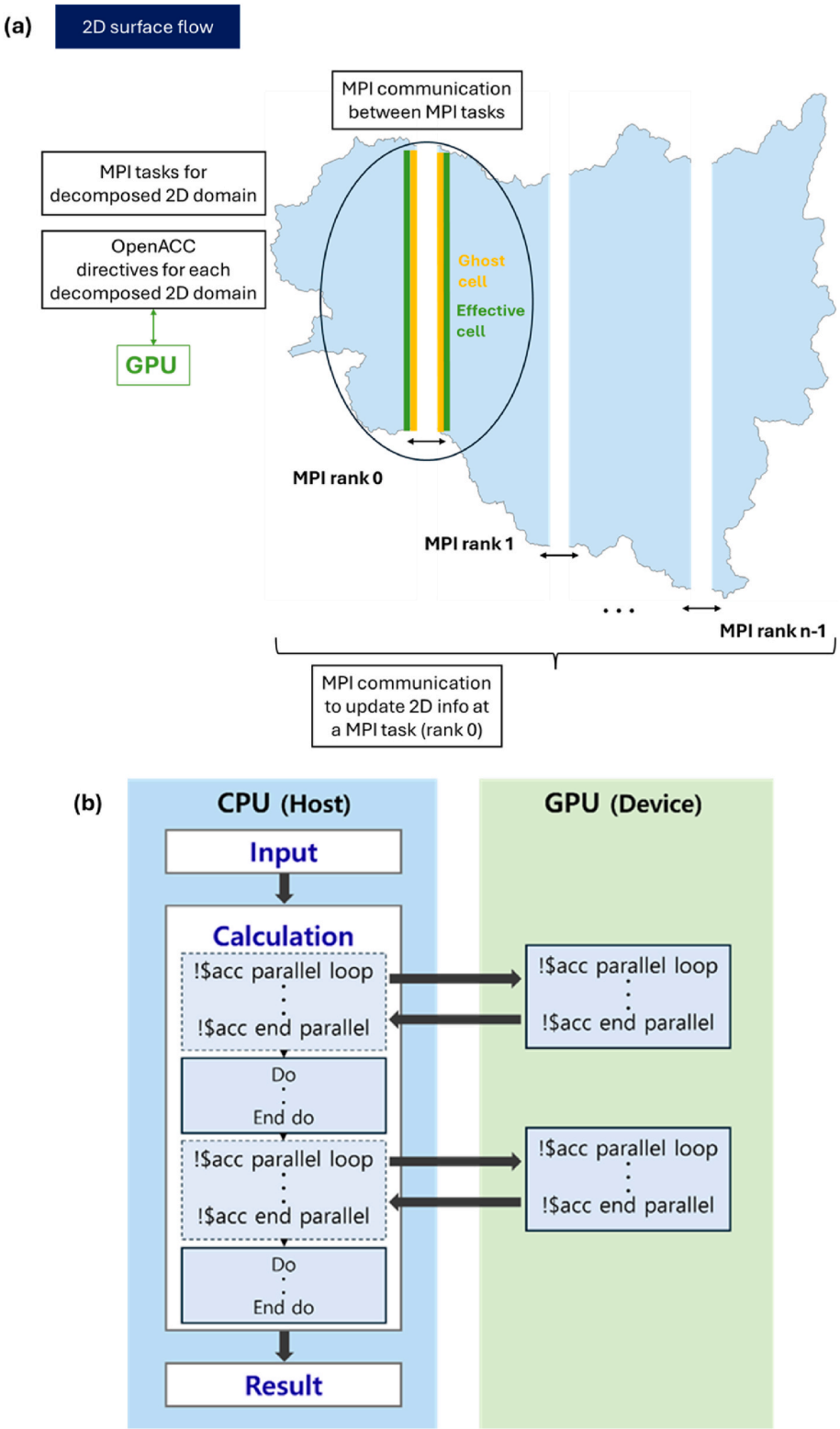

Fig. 1. Schematic diagram of the MPI–OpenACC implementation for 2D surface flow computation: (a) 1D slab domain decomposition with ghost and effective cells exchanged between neighboring MPI ranks; (b) CPU–GPU execution workflow showing OpenACC parallel loop offloading of interior stencil updates and host-side coordination of halo exchange.

with each rank offloading compute-intensive loops to a GPU via OpenACC. At subdomain interfaces, neighboring ranks exchange halo values using non-blocking point-to-point communication. Ghost cells receive updated boundary states from adjacent ranks, and effective cells are advanced by device kernels. This configuration overlaps communication with interior computation, facilitating scalable execution on multiple GPUs. While MPI-OpenACC multi-GPU implementations have been reported in both engineering solvers and selected flood-modelling applications, the present work emphasises performance portability within an existing operational hydrological framework and evaluates scalability across heterogeneous basin configurations and multiple spatial resolutions. This focus on workload-per-GPU synthesis under realistic urban modelling conditions distinguishes the present study from prior implementations. Fig. 1b illustrates the hybrid execution workflow. Interior stencil loops are offloaded to the GPU using OpenACC parallel loop directives, while the host coordinates time stepping and MPI halo exchange between neighboring ranks. This structure enables the overlap of device computation with inter-rank communication.

Fig. 2a presents the profiling data from NVIDIA Nsight Systems for a GPU execution on a Tesla V100 PCIe board equipped with 32 GB of memory. Nsight Systems captures kernel timelines and memory activity. In our GPU-accelerated program, device kernels constitute approximately 78.7% of the runtime, while unified memory activity accounts for about 21.3%. The kernel time is predominantly occupied by flux_43_gpu at 36.3%, which advances the momentum equations on surface grids, and suisin_52_gpu at 27.5%, which advances the continuity equation on surface grids. Subsequent auxiliary updates include mdvel_27_gpu at 14.7%, mdvel_157_gpu at 10.2%, mdvel_119_gpu at 5.8%, and the time loop driver forward_20_gpu at 5.5%. The memory section reveals 59.9% host-to-device transfers and 40.1% device-to-host transfers. Fig. 2b outlines the module flow in the GPU-accelerated 2D urban flood model and the relative runtime of each module within the time loop. State arrays are primarily accessed by device kernels during the time loop. Momentum and continuity kernels update interior cells, while non-blocking MPI exchanges halo values to facilitate the overlap of communication and computation. Unified memory migrates accessed pages between host and device as needed. Because halo exchange is performed on the host, page migration may occur during boundary synchronisation. The unified memory activity observed in profiling therefore reflects page-level migration associated with halo exchange rather than explicit full-domain copies at every time step.

2.3. Evaluation metrics

Speedup (S) is computed as the ratio of the serial runtime (T_{serial}) to the parallel runtime ($T_{parallel}$) for all runs.

$$S = \frac{T_{serial}}{T_{parallel}} \tag{5}$$

Efficiency for the CPU-hybrid (E_{CPU}) is computed as speedup divided by the number of CPU cores (N_{core}). Efficiency for the GPU hybrid (E_{GPU}) is normalised to the 2 GPU case so that the 2 GPU configuration ($S_{2\ GPU}$) is 1 and the 4 GPU configuration ($S_{G\ GPU}$) reflects scaling relative to that baseline. This definition represents a relative scaling metric referenced to the 2 GPU configuration and is distinct from the conventional scaling efficiency based on ideal linear speedup from a single-device baseline. G is GPU counts, CPU and GPU efficiency is computed as

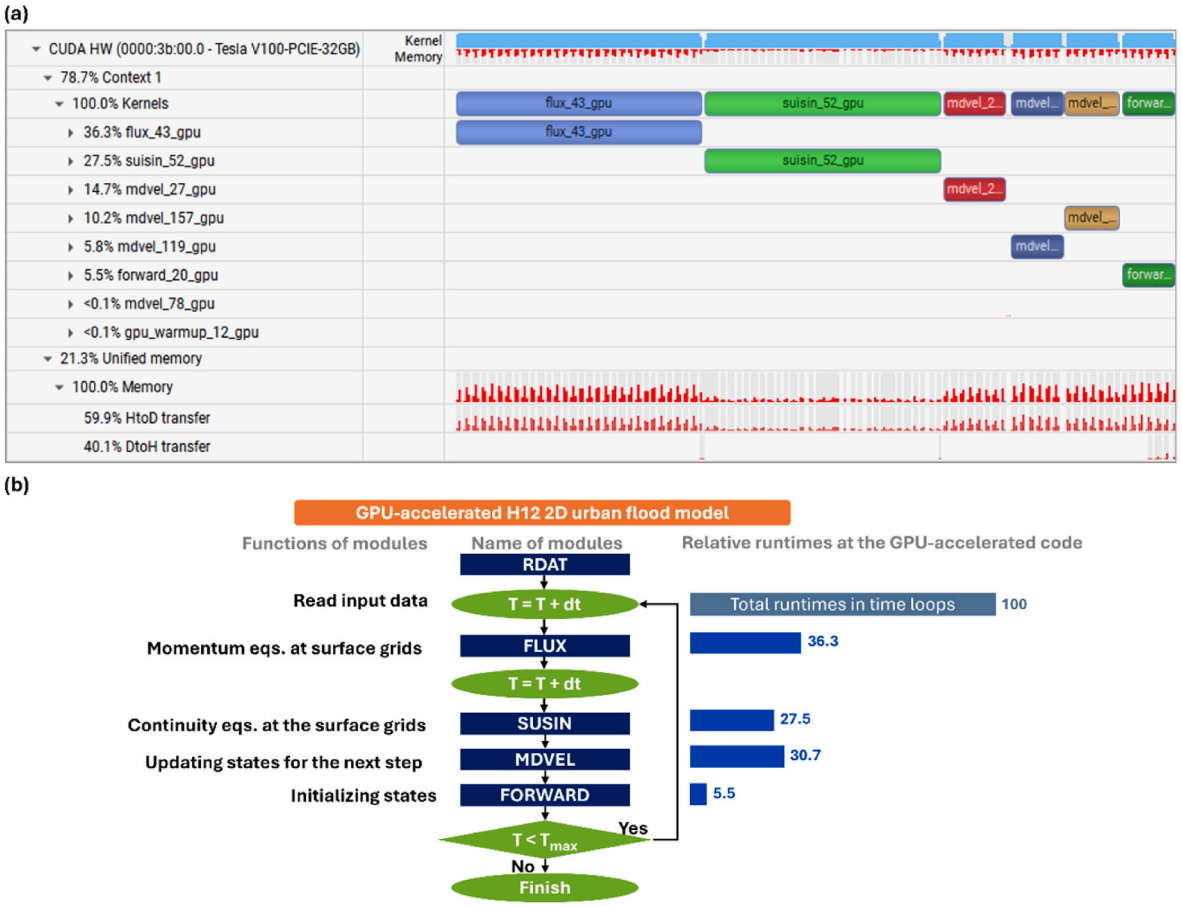

Fig. 2. Profiling and module runtimes for the MPI and GPU hybrid urban flood model: (a) Nsight Systems kernel timeline on a Tesla V100; (b) model workflow with relative time shares.

$$E_{CPU} = \frac{S}{N_{core}} \tag{6}$$

$$E_{GPU} = \frac{S_{G\ GPU}/S_{2\ GPU}}{G/2} \tag{7}$$

3. Study area and experimental setup

To assess applicability across contrasting settings, we applied the model to two study basins. The Portland basin in United States, basin was chosen to represent pluvial-only conditions, while the downstream Han River basin in South Korea was selected to represent concurrent pluvial and fluvial flooding.

3.1. Downtown Portland

The study area selected for simulating pluvial urban flooding is located in the western portion of downtown Portland, Oregon, USA, as depicted in Fig. 3. This basin is situated on the west bank of the Willamette River within the western Willamette Basin. The mean annual precipitation is approximately 930 mm (Lee et al., 2025). The Portland basin is relatively small, encompassing 9.0 km^2, with a gentle topography and a total elevation difference of only 57 m. The area's diverse topography and intricate urban structure, combined with a variety of land-cover types, render it an appropriate testbed for assessing urban inundation models. In this study, we employed the 100-year design rainfall for Portland as specified in NOAA Atlas 2(see https://www.weather.gov/owp/hdsc_noaa_atlas2). The analysis considers only 2D surface runoff, with storm-sewer networks not explicitly modelled. A 1 m resolution digital elevation model, derived from LiDAR data obtained via the National Oceanic and Atmospheric Administration Data Access Viewer, was utilised. To incorporate urban structural features, building heights were added to the base DEM to create a digital surface model. Building footprints with height attributes were sourced from Portland Maps Open Data (see https://gis-pdx.opendata.arcgis.com/), which provides open geospatial layers for the city. As only buildings were included, artificial structures such as bridges were excluded from the digital surface model. For the land cover used to assign runoff coefficients, we utilised the metre-scale urban land cover dataset from the United States Environmental Protection Agency's EnviroAtlas (Pickard et al., 2015). Of the eight mapped classes, including water, impervious surfaces, and wetlands, four are present in the study basin: water, impervious surfaces, soil and barren or non-productive land, and trees and forest. Runoff coefficients for each land cover class were based on prior research (Lee et al., 2025). The land cover fractions are 65.6% impervious surfaces, 31.0% vegetation and bare land, and 3.3% water bodies, indicating a high degree of urban imperviousness. Consequently, basin response times are brief, runoff peaks develop rapidly, and contributions from river inflow are comparatively minor.

3.2. Downstream Han River basin

The study area encompasses the downstream Han River basin in South Korea, selected to exemplify the confluence of pluvial and riverine flooding (Fig. 4). This basin extends from the downstream reach of the dam to the Han River estuary, covering an area of 2386 km^2. It includes the entire Seoul Metropolitan Area, Incheon Metropolitan City, and portions of Gyeonggi Province. As of early 2025, Seoul's population is approximately 9.60 million, while Incheon has about 3.03 million residents. Even without considering the entirety of Gyeonggi Province, these two cities alone suggest a potentially affected population exceeding 12.6 million, and the inclusion of the Gyeonggi portion significantly increases this figure. The Han River main stem, traversing this basin, is one of South Korea's four major rivers and receives inflow from numerous tributaries, including the Jungnangcheon, Tancheon, Hongjecheon, and Anyangcheon streams. This study focuses solely on 2D surface runoff, with storm sewer networks not explicitly modelled.

To construct the Digital Elevation Model (DEM), 1 m digital elevation data from the National Geographic Information Institute of Korea were compiled and resampled to 10, 20 m. River cross sections from the Han River SWMM dataset were extracted as geospatial features and incorporated into the DEM to create a hydrologically conditioned surface representing channel geometry. For the Han River estuary, where NGII data were unavailable due to security reasons, the Shuttle Radar Topography Mission Global 30 m product was utilised, resampled to 10, 20 m, and cross sections were linearly interpolated prior to integration. To represent urban drainage, it was assumed that the storm sewer network conveys flows up to its design capacity without restriction, and surface inundation was simulated only for excess rainfall beyond that capacity. Runoff coefficients necessary for the runoff simulation were assigned based on land cover class. The land cover map was produced by

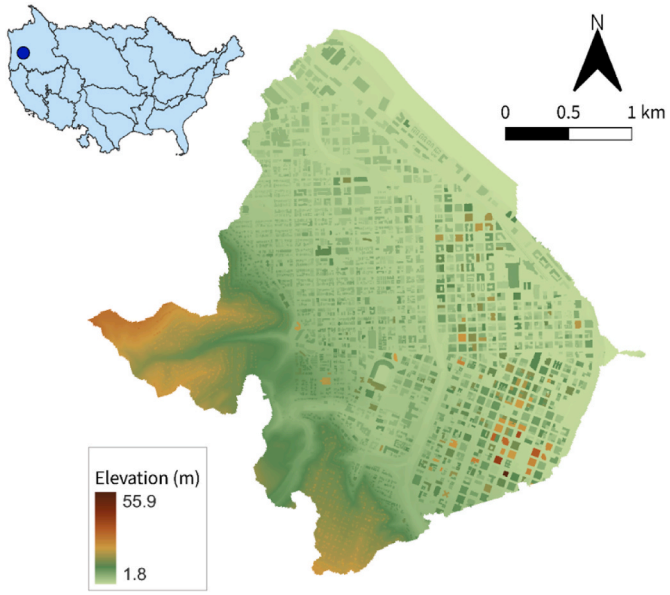

Fig. 3. 1 m spatial resolution DSM of Portland area.

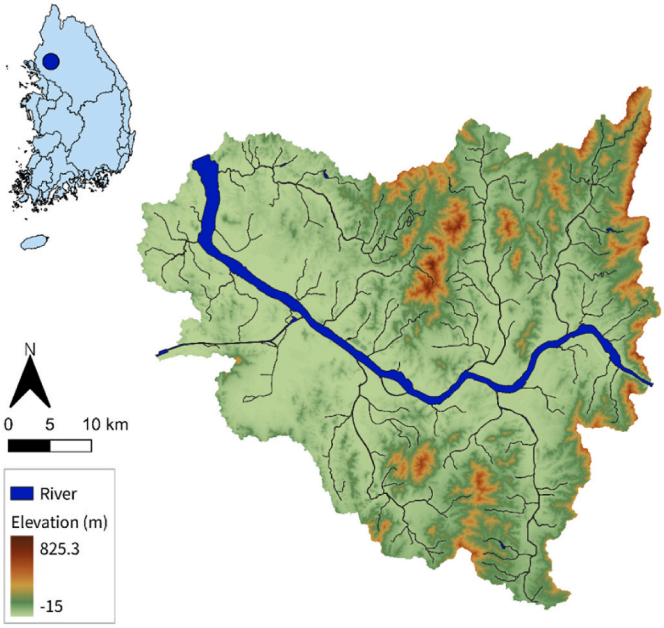

Fig. 4. 20 m spatial resolution DEM of Han River downstream area.

reclassifying the Korean Ministry of Environment's 5 m, 22-class mid-level land cover dataset into the four classes used in the model: buildings, roads, water, and other areas.

Land cover fractions are 32.7% impervious surfaces, 63.1% vegetation and bare land, and 4.2% water bodies, indicating the predominance of vegetated and bare surfaces. The basin's scale and relief facilitate the simultaneous influence of intense rainfall-generated overland flow and stage variations transmitted from upstream river reaches.

3.3. Benchmark experimental setup

To verify the model, it was applied to two basins with distinct spatial coverage and hydrological characteristics. The downtown Portland basin in the United States exemplifies pluvial-only conditions, whereas the downstream Han River basin in South Korea represents conditions where rainfall-runoff and upstream river inflow can occur simultaneously. A summary of physical characteristics, land cover, and grid configuration is provided in Table 1. The Portland basin is characterized as an urban environment with high imperviousness and low relief, dominated by pluvial processes, while the Han River basin encompasses a large area with higher relief and a significant river influence.

Table 2 summarises the key simulation settings for both basins, including simulation duration, forcing type, rainfall temporal resolution, grid resolutions tested, and boundary inflow conditions. The computational grid was customised for each basin's specific requirements. For the Portland basin, resolutions of 1, 2, and 4 m were employed to capture fine-scale urban topography. In contrast, for the Han River basin, resolutions of 10 and 20 m were utilised to accommodate the extensive spatial coverage. This selection of spatial scale and resolution was intended to rigorously test the proposed MPI-GPU hybrid solver under varying computational demands. All tests utilised the same input data, source code, and numerical settings, with variations only in the CPU/GPU configuration.

In the downtown Portland basin application, a 100-year return-period rainfall event was simulated for a duration of 6 h, generated as a 5 min interval time-series hyetograph. For the large-domain downstream Han River basin application, which represents concurrent pluvial and fluvial forcing, a 500-year design rainfall event was simulated for 24 h. The spatial rainfall field was prepared at a 50 m resolution using regional frequency analysis and the Thiessen polygon method. To represent riverine overflow, initial conditions were established by applying a constant boundary inflow equivalent to the 500-year discharge of 41,532 m^3/s over a spin-up period, with the 24 h simulation proceeding with the same inflow maintained as a boundary condition throughout. For the Han River basin, a DEM was used instead of a DSM, so building heights are not embedded in the grid. Post-simulation, the depth field was processed with building footprints from the National Geographic Information Institute of Korea to ensure inundation follows the digital surface of the built environment.

Table 1
Summary of basin characteristics.

Study area		Portland basin	Downstream Han River basin
Elevation difference		57 m	840 m
Land cover	Impervious areas	65.6%	32.7%
	Vegetation & bare land	31.0%	63.1%
	Water body	3.3%	4.2%
Area		9.0 km^2	2386 km^2
Spatial resolution (m)		1, 2, 4	10, 20
No. of grid		1 m: 9,009,045 2 m: 2,252,235 4 m: 1,000,988	10 m: 23,855,776 20 m: 5,972,792

Table 2
Summary of simulation configurations for benchmark cases.

Item	Portland basin	Downstream Han River basin
Simulation periods	6 h (full simulation) 1 h (benchmark runs)	24 h (full simulation) 1 h (benchmark runs)
Flooding type	Pluvial only	Pluvial + fluvial
Temporal resolution of rainfall forcings	5 min	1 h
Grid resolutions tested	1 m, 2 m, 4 m	10 m, 20 m
Boundary inflow setting	No upstream inflow imposed	Constant inflow equivalent to 500-year discharge (41,532 m^3/s)
Downstream boundary condition	No downstream boundary imposed	Constant water level set to 7.4 m

Benchmark tests were conducted as 1 h simulations for each basin under varying computational resource configurations. The benchmarks encompassed three scenarios: a serial execution on a single CPU core, a CPU-hybrid configuration utilising MPI and OpenMP, and a GPU-hybrid configuration employing MPI and OpenACC on GPUs. All three scenarios yielded identical inundation results, with the concordance between the CPU-hybrid and GPU-hybrid configurations for the Portland basin illustrated in Fig. S2. Numerical consistency was assessed using L_1, L_2 and L_∞ norms over wet cells at each output time, and for the cases compared the depth fields matched to three decimal places (0.001 m) on the common grid. While GPU architectures often achieve higher throughput with single precision, double precision was retained in this study to maintain consistency with existing H12 applications. To benchmark rainfall, the 100-year event for Portland was used, while for the downstream Han River basin, an hourly intensity of 150 mm/h was set, surpassing the 500-year intensity for Seoul of 125.11 mm/h. This selection reflects the severe event on August 8, 2022, when Mokdong in Seoul recorded 141.5 mm/h.

The GPU-based simulations were executed on a GPU-only node at the Korea Institute of Science and Technology Information. The system employed was an HPE Cray XD670 equipped with eight NVIDIA H200 GPUs, each providing a theoretical peak memory bandwidth of approximately 4.8 TB/s via HBM3e memory. The GPUs are interconnected through NVLink within the HGX baseboard to enable high-bandwidth GPU-to-GPU communication. The associated host processor was an Intel Xeon Emerald Rapids with 48 cores using DDR5 memory architecture. Given that this machine was exclusively GPU-based, CPU-only experiments were conducted on a separate system. The CPU-only configuration consisted of five nodes, each equipped with two Intel Xeon Gold 6248R processors, providing 48 cores per node. A link to sample runtime benchmark results for both basins is provided in the Data Availability section. The CPU-hybrid configuration was compiled using the Intel MPI Fortran compiler with -O2 optimisation and OpenMP enabled. The GPU-hybrid configuration was compiled using the NVIDIA HPC SDK with OpenACC enabled, targeting compute capability 9.0 with unified memory support. MPI communication employed a CUDA-aware implementation.

4. Results

In this section we apply and evaluate the proposed framework across two basins with contrasting characteristics.

4.1. Portland city-scale basin case

We applied the multi-GPU framework to the Portland basin for a 100-year, 6 h design storm at 1 m resolution (Fig. 5). Inundation expands through hours 3, 4 and then recedes, with deeper water along waterfront margins, broad streets, and enclosed yards where local gradients limit drainage. Inundation patterns align with the digital surface model:

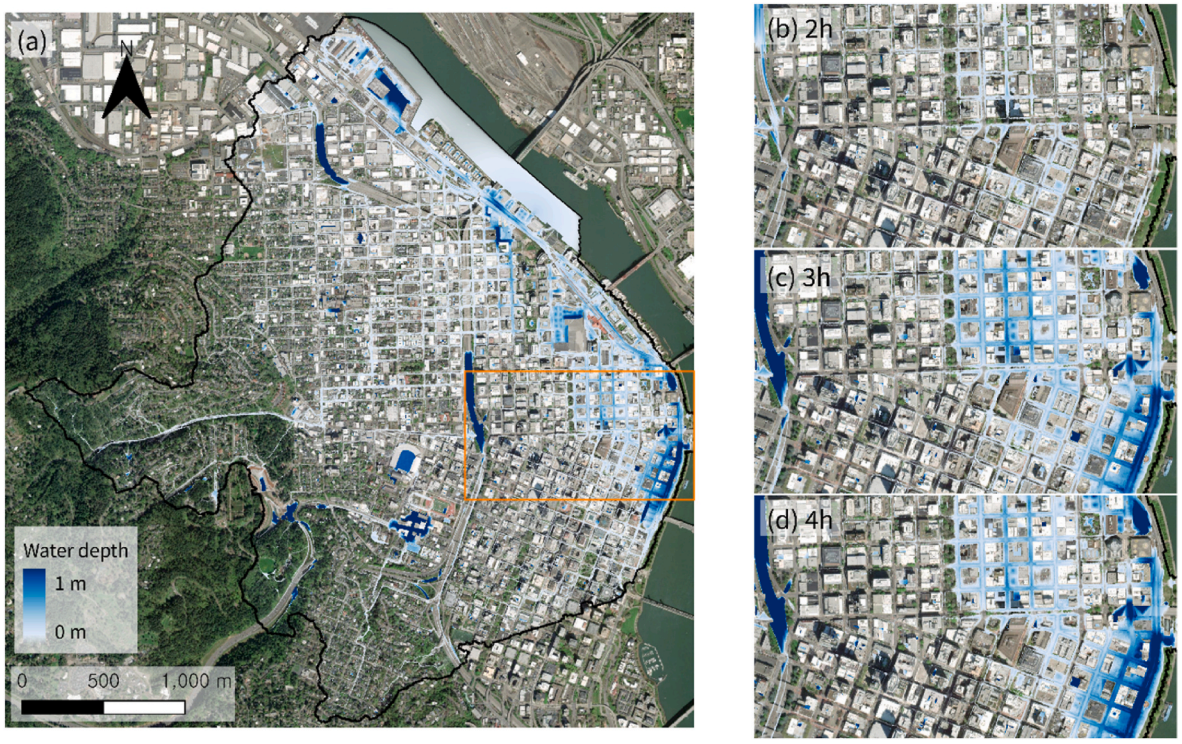

Fig. 5. Hourly inundation maps for the Portland basin under the 100-year, 6 h event at 1 m resolution. Panels show 2-4 h with enlarged views; the color scale indicates water depth (0-1 m). The full 1 m hourly sequence (1-6 h) is provided in Fig. S3. (For interpretation of the references to color in this figure legend, the reader is referred to the Web version of this article.)

building blocks steer overland routing along road corridors and between structures, while small depressions act as temporary storage. By hour 4 the areal extent begins to contract as surface storage drains and shallow ponding diminish. The complete 1 m hourly sequence is shown in Fig. S3. Model validation and the analysis of urban flood results across spatial resolutions were discussed in Lee et al. (2025).

In evaluating performance, we conducted benchmark tests with independent 1 h runs at resolutions of 1, 2, and 4 m (Fig. 6; Table 3). The CPU-hybrid configuration utilised 144 cores (3 nodes × 8 MPI ranks × 6 OpenMP threads). Across all resolutions, the GPU-hybrid configuration demonstrated superior performance compared to the CPU-hybrid. At a resolution of 1 m (9,009,045 cells), the optimal runtime was 105.3 s on four GPUs, representing a speedup of 407.8 times compared to the serial baseline (42,943.1 s) and 13.5 times faster than the 144-core CPU-hybrid (1424.4 s). At 2 m (2,252,235 cells), two GPUs achieved a runtime of 47.5 s, resulting in a speedup of 207.8 times over the serial configuration and 8.5 times over the CPU-hybrid. At 4 m (1,000,988 cells), two GPUs achieved a runtime of 23.4 s, equating to a speedup of 112.0 times over the serial configuration and 6.5 times over the CPU-hybrid. As illustrated in Table 3, the scaling trend is influenced by the workload per device. For the 1 m grid, the runtime decreased from 137.5 s on two GPUs to 105.3 s on four GPUs (a reduction of approximately 24 percent) but increased to 114.9 s on eight GPUs as halo exchanges and data movement began to counteract the added computational capacity. For the 2 and 4 m grids, two GPUs provided the fastest runtimes; adding more devices increased the runtime because communication and synchronisation costs became dominant when the workload per device was reduced. On this hardware–model

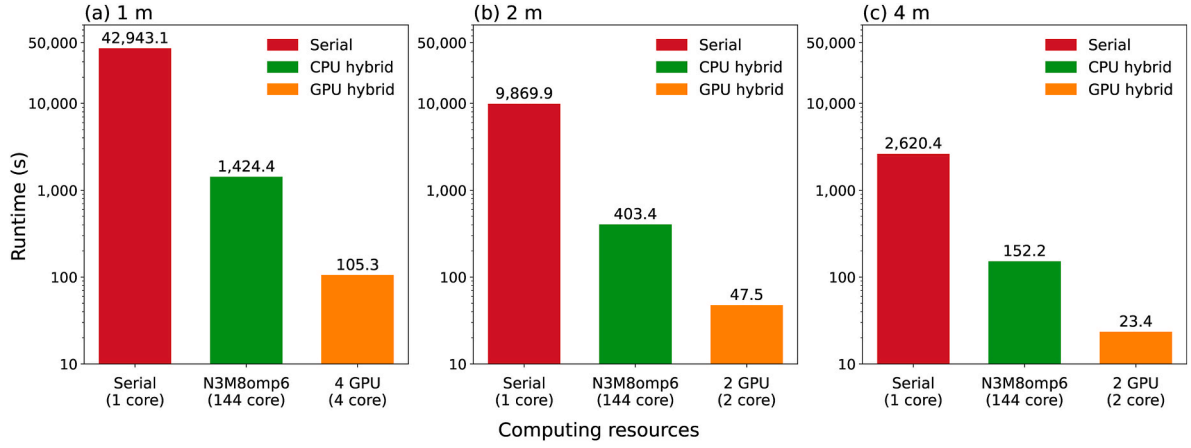

Fig. 6. Runtime and speedup for the Portland benchmarks at 1, 2, and 4 m resolutions. Bars show wall-clock runtime; markers show speedup relative to the serial baseline. The CPU-hybrid uses N3M8omp6 (3 nodes × 8 MPI ranks × 6 threads = 144 cores). For each resolution, the GPU-hybrid bar reflects the best runtime across tested device counts (4 GPUs at 1 m; 2 GPUs at 2 m and 4 m). Exact values are listed in Table 3.

Table 3
Wall-clock runtime (s) for 1 h Portland simulations under 100-year rainfall at 1, 2, and 4 m. Serial = 1 core; CPU-hybrid = N3M8omp6 (144 cores); GPU-hybrid results shown for 2/4/8 GPUs. Values correspond to Fig. 6.

Resolution	No. grid	Resources		Runtime
1 m	9,009,045	No. CPU	Serial (1 core)	42943.1
			N3M8omp6 (144 core)	1424.4
		No. GPU	2	137.5
			4	**105.3**
			8	114.9
2 m	2,252,235	No. CPU	Serial (1 core)	9869.9
			N3M8omp6 (144 core)	403.4
		No. GPU	**2**	**47.5**
			4	50.7
			8	75.2
4 m	1,000,988	No. CPU	Serial (1 core)	2620.4
			N3M8omp6 (144 core)	152.2
		No. GPU	**2**	**23.4**
			4	45.2
			8	78.3

combination, optimal scaling is achieved when each GPU processes approximately 1 to 2.5 million surface cells. When the workload falls below approximately 0.6 million cells per GPU, efficiency declines, and additional devices do not reduce the time to solution. This behavior is consistent with previous GPU studies in environmental modelling, which report declining efficiency at high device counts as communication and synchronisation costs increase relative to computation (Morales Hernández et al., 2021; Saleem and Norman, 2024; Dong et al., 2025b). A complementary scaling analysis for a larger-scale domain is presented in the following subsection.

4.2. Downstream Han River (large-domain) case

Building on the city-scale analysis in Section 4.1, we assess a larger riverine domain to examine how scale and grid resolution affect both inundation patterns and multi-GPU performance.

4.2.1. Inundation patterns and hotspots

We evaluated the framework on the downstream Han River basin with a resolution of 20 m during a 500-year, 24 h event (Fig. 7). The simulation indicates the deepest water along the main stem and adjacent floodplains, reflecting fluvial control from the imposed river release, while inundation distant from the river is primarily influenced by pluvial accumulation from design rainfall and limited internal drainage. Ponding in interior areas corresponds with terrain convergence lines and local flow paths. The insets in Fig. 7 emphasise recurring controls: interior ponding over low-lying blocks in Dapsimni (Fig. 7b), backwater-enhanced storage near the Anyangcheon confluence in Mokdong (Fig. 7c), and a combination of confluence influence and accumulation over low-relief reclaimed ground in Jamsil (Fig. 7d). Depth gradients intensify along embanked margins and inner bends, consistent with compound forcing from external river levels and terrain-guided accumulation.

4.2.2. Runtime scaling on large-domain grids

We conducted independent 1 h benchmarks at resolutions of 10 m (23.86 million cells) and 20 m (5.97 million cells) under identical 500-year rainfall forcing conditions (Fig. 8; Fig. 9). Across both resolutions,

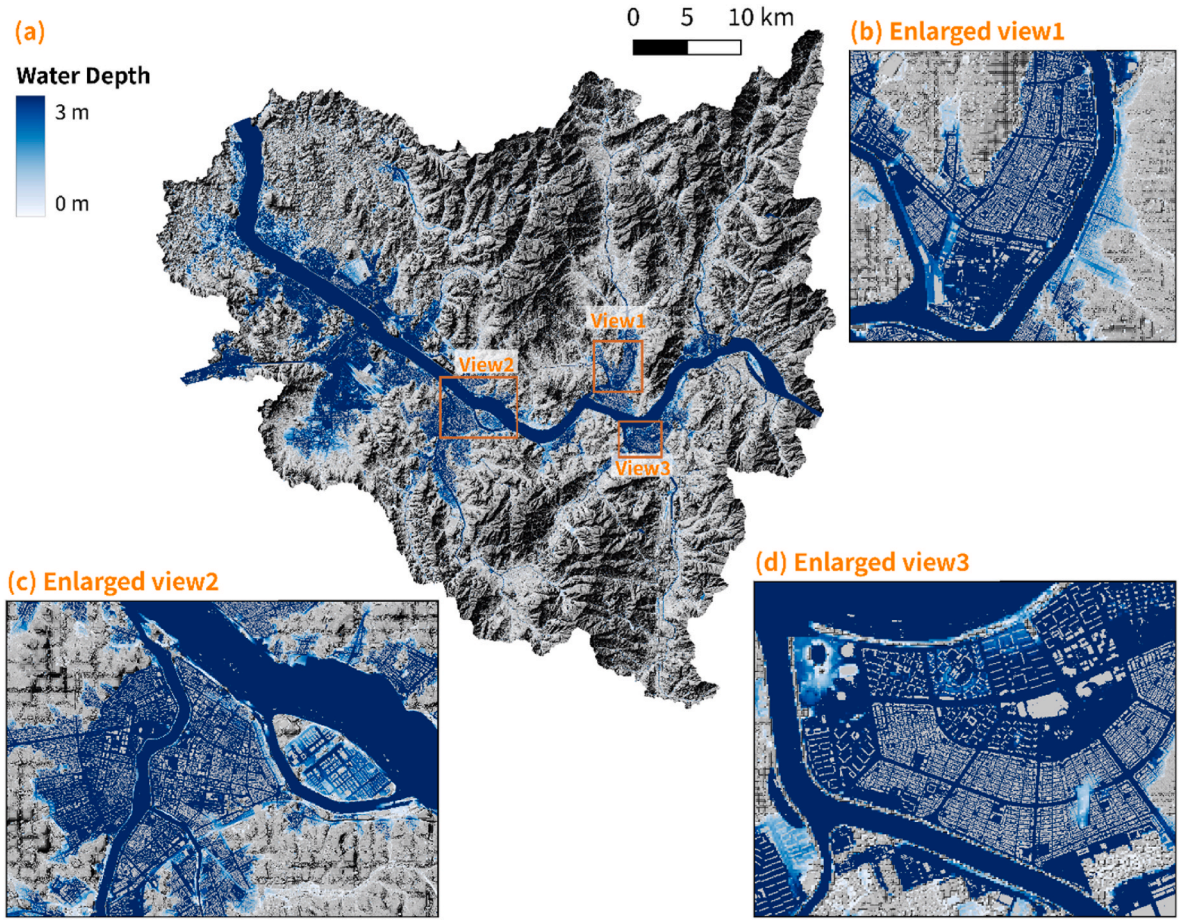

Fig. 7. Maximum inundation depth in the downstream Han River basin for the 500-year, 24 h event. Insets (views 1–3) provide enlarged water depth maps for Dapsimni (b), Mokdong (c), and Jamsil (d) to clarify local patterns.

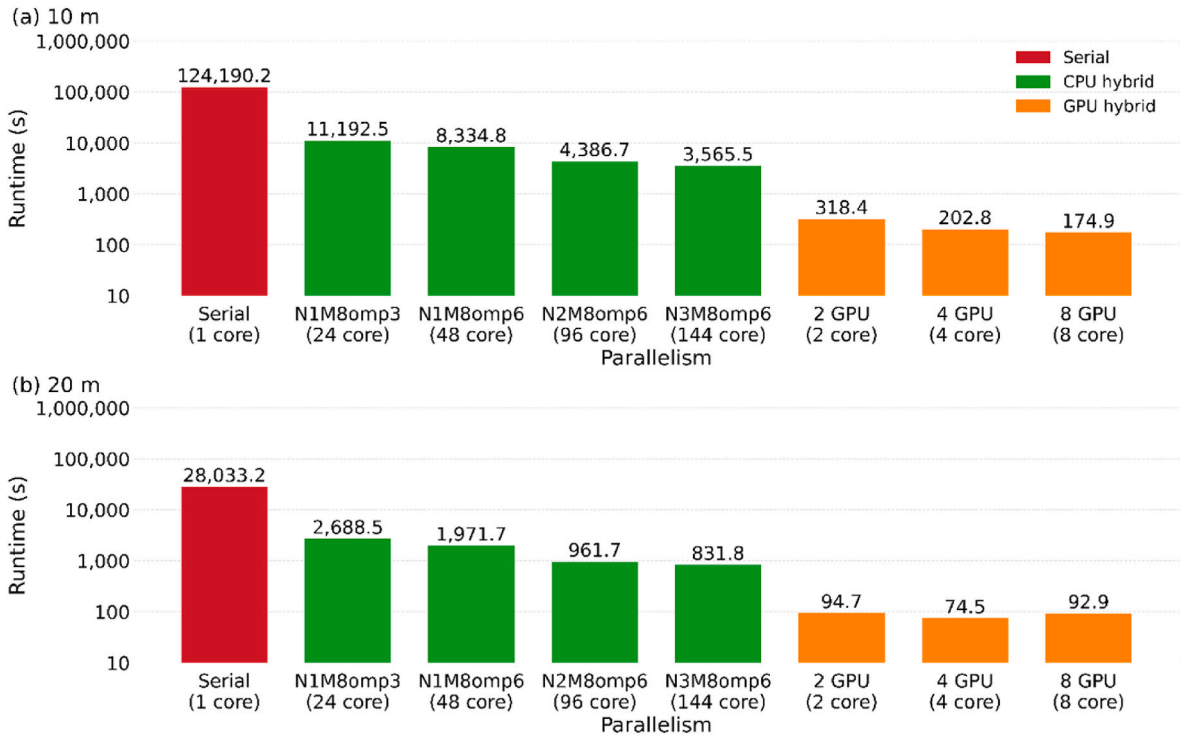

Fig. 8. Comparison of serial, CPU-hybrid, and GPU-hybrid runtimes for 1 h downstream Han River simulations forced by 500-year design rainfall at 10 m and 20 m resolutions.

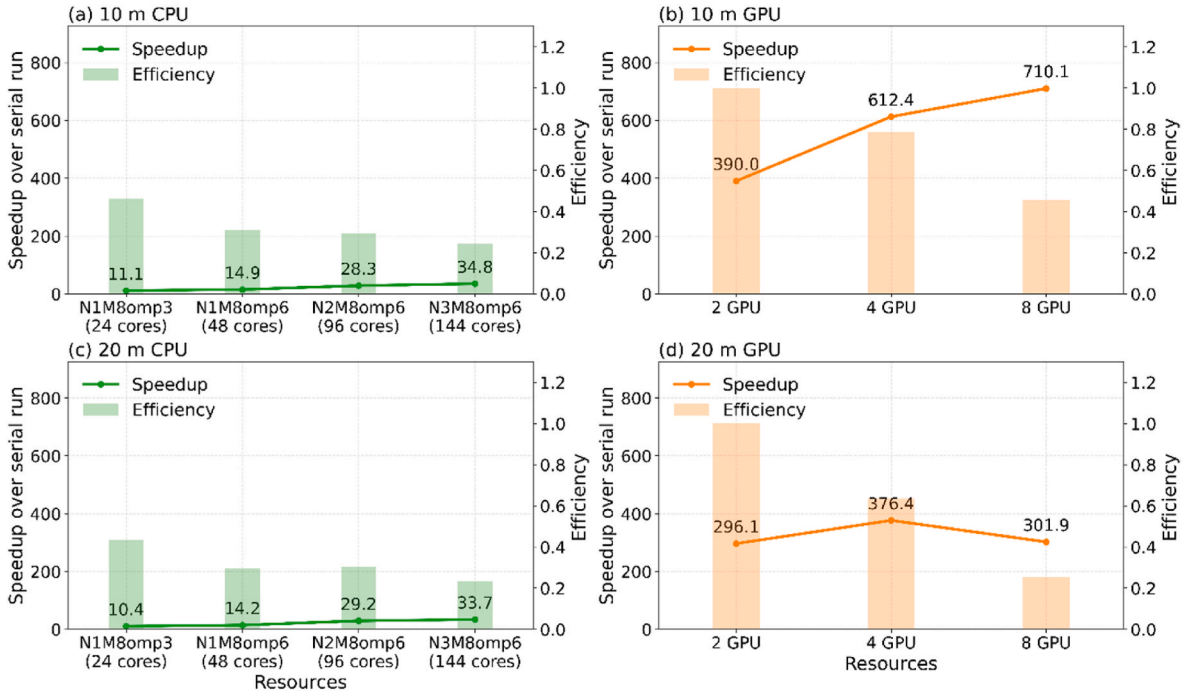

Fig. 9. Speedup and efficiency for CPU-hybrid (a, c) and GPU-hybrid (b, d) runs in the downstream Han River basin with 10 and 20 m resolutions. GPU efficiency is defined relative to the 2-GPU configuration (normalised to 1) and represents scaling performance rather than absolute parallel efficiency.

the GPU-hybrid configuration demonstrated superior performance compared to the CPU-hybrid configuration. These results are not a direct hardware comparison between CPU and GPU platforms.

In the analysis of wall-clock runtime (Fig. 8), we executed independent 1 h runs at 10 m with 23.86 million cells and at 20 m with 5.97 million cells under the same 500-year rainfall forcing (Fig. 8). At the 10 m resolution, the serial baseline required 124,190.2 s; the CPU-hybrid reduced this to 3565.5 s using 144 cores, while the GPU-hybrid achieved 318.4 s on 2 GPUs, 202.8 s on 4 GPUs, and 174.9 s on 8 GPUs. At the 20 m resolution, the serial baseline required 28,033.2 s; the CPU-hybrid reached 831.8 s with 144 cores, and the GPU-hybrid achieved 94.7 s on 2 GPUs, 74.5 s on 4 GPUs, and 92.9 s on 8 GPUs. The optimal

time-to-solution was thus observed at 8 GPUs for the 10 m resolution and at 4 GPUs for the 20 m resolution, indicating that the optimal number of devices is contingent upon the workload per GPU.

Regarding speedup and efficiency (Fig. 9), the 10 m case reveals that the CPU-hybrid speedup increased from 11.1 × at 24 cores to 34.8 × at 144 cores, with efficiency declining from approximately 0.46 to 0.24. The GPU-hybrid attained a speedup of 390.0 × at 2 GPUs, 612.4 × at 4 GPUs, and 710.1 × at 8 GPUs; efficiency decreased from 1.00 at 2 GPUs to 0.79 at 4 GPUs and 0.45 at 8 GPUs. In the 20 m case, the CPU-hybrid speedup increased from 10.4 × at 24 cores to 33.7 × at 144 cores, with efficiency declining from approximately 0.43 to 0.23. The GPU-hybrid achieved a speedup of 296.1 × at 2 GPUs, 376.4 × at 4 GPUs, and 301.9 × at 8 GPUs; efficiency changed from 1.00 at 2 GPUs to 0.64 at 4 GPUs and 0.26 at 8 GPUs. Collectively, these results suggest that effective scaling is achieved when each GPU processes approximately 1 to 2.5 million cells; when the workload per GPU decreases to below one million cells, the increasing proportion of halo exchanges and synchronisation offsets additional computation, consistent with the thresholds identified in Section 4.1.

Beyond the realm of urban flood modelling, GPUs are increasingly being integrated into various disciplines such as weather and ocean modelling. In the domain of weather prediction, the CUDA ports of the Weather Research and Forecasting (WRF) model's microphysics and an OpenACC port of the Model for Prediction Across Scales (MPAS) microphysics have demonstrated significant speed enhancements while maintaining verified accuracy (Mielikainen et al., 2013; Huang et al., 2015; Kim et al., 2021). In ocean modelling, both full GPU implementations and directive-based approaches have shown comparable performance improvements, as evidenced by POM.gpu and a recent OpenACC implementation of the Princeton Ocean Model (POM), as well as large-scale GPU deployments of the LASG/IAP Climate System Ocean Model (LICOM3) utilising HIP and performance-portable Kokkos (Huang et al., 2015; Wang et al., 2021, 2025). Efforts on the Nucleus for European Modelling of the Ocean (NEMO) also target GPUs through OpenACC and PSyclone code transformation, with recent practices emphasising unified memory workflows (Porter and Heimbach, 2025). Collectively, these cross-domain findings underscore the critical importance of aligning workload with device count and managing communication costs, suggesting that the cells per GPU guidance from this study is likely applicable beyond urban inundation scenarios.

5. Conclusions

This study presented a portable, directive-based multi-GPU implementation of the H12 2D urban flood model using MPI-OpenACC and evaluated it across a metre-resolution city case and a large-domain basin case. A single code base supports CPU-only and GPU-accelerated execution through compile-time options. The main findings are as follows:

- The model reproduced identical inundation fields between CPU-hybrid and GPU-hybrid runs for the Portland basin, confirming numerical consistency while delivering large runtime reductions.
- In the Portland application the GPU hybrid achieved about 408 × at 1 m, about 208 × at 2 m and about 112 × at 4 m relative to the serial baseline and remained faster than the CPU hybrid at all resolutions.
- In the downstream Han River basin the fastest runtime occurred with 8 GPUs at 10 m, giving about 710 × over the serial baseline; at 20 m the optimum shifted to 4 GPUs (about 376 ×), confirming that the best device count depends on mesh size and that cells-per-GPU matching is critical, with solution consistency preserved across CPU and GPU runs.
- Resolution scaling revealed a clear relationship between grid size and the optimal number of GPUs; for this model and hardware, good scaling occurs when each GPU carries between 1 and 2.5 million surface cells.

Together, these findings show that a directive-based multi-GPU design delivers order-of-magnitude runtime reductions without loss of fidelity and yields a practical provisioning guideline: maintain 1-2.5 million active surface cells per GPU (with diminishing returns below 0.6 million). With this foundation, the dominant remaining bottlenecks are interconnect latency and host-device transfers, which motivate the following directions. Future research could evaluate deployment on clusters with multi-node NVLink, including NVL72-class systems, to reduce halo-exchange latency and extend strong scaling for large domains (NVIDIA, 2025a, 2025b). Another avenue involves maintaining halo buffers and model state fully resident on the device across time steps using OpenACC data regions with the present clause. This approach could further reduce unified memory page migration associated with host-side halo exchange, increase overlap between communication and computation, and lower runtime without affecting solution fidelity (OpenACC, 2023; NVIDIA, 2025c). The accelerated framework could also be extended from 2D surface flow to coupled 1D–2D simulations so that river channels and urban drainage networks are represented alongside overland flow.

CRediT authorship contribution statement

Bomi Kim: Writing – review & editing, Writing – original draft, Methodology. **Hyungon Ryu:** Writing – review & editing, Software, Methodology. **Seungsoo Lee:** Writing – review & editing, Validation. **Jun-Hak Lee:** Writing – review & editing, Data curation. **Seong Jin Noh:** Writing – review & editing, Supervision, Methodology, Funding acquisition, Conceptualization.

Software availability

The CPU version of the H12 2D urban flood model, together with the 4 m resolution sample input dataset for the Portland test case, CPU/GPU output results corresponding to Fig. S2, and runtime benchmark logs for the downstream Han River and downtown Portland experiments, is archived on Zenodo (DOI: https://doi.org/10.5281/zenodo.18627484). The GPU-accelerated version of the code is not publicly available.

- Name of software: H12 2D urban flood model
- Developers: Bomi Kim and Seong Jin Noh
- Contact: kimbom3835@gmail.com and seongjin.noh@gmail.com
- Date first available: Feb 13, 2026
- Software required: Linux, MPI library
- Program language: Fortran 90
- Source code at: https://doi.org/10.5281/zenodo.18627484

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported by the Climate Resilient R&D Project for Water-Related Disaster Management funded by the Korea Ministry of Climate, Energy and Environment and implemented by the Korea Environment Industry & Technology Institute (KEITI) (RS-2023-00218973); by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. RS-2023-00246532); and by the Technology Innovation Program (RS202400398858, Development of AI-based urban flood damage risk prediction and evaluation technology for practical use) funded by the Ministry of the Interior and Safety (MOIS, Korea).

We thank Professor Juyoung Shin (Kookmin University, Republic of Korea) for providing probabilistic design rainfall data for the

downstream Han River basin. We also acknowledge computing resources were provided by the KIT Supercomputing Center at Kumoh National Institute of Technology and by the National Supercomputing Center at KISTI.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.envsoft.2026.106957.

Data availability

The source code, input datasets, simulation results, and modelling runtime logs have been deposited in a Zenodo repository, and the manuscript includes the corresponding links and information.

References

Buttinger-Kreuzhuber, A., Konev, A., Horváth, Z., Cornel, D., Schwerdorf, I., Blöschl, G., Waser, J., 2022. An integrated GPU-accelerated modeling framework for high-resolution simulations of rural and urban flash floods. Environ. Model. Software 156, 105480. https://doi.org/10.1016/j.envsoft.2022.105480.

Carlotto, T., Borges Chaffe, P.L., Innocente dos Santos, C., Lee, S., 2021. SW2D-GPU: a two-dimensional shallow water model accelerated by GPGPU. Environ. Model. Software 145, 105205. https://doi.org/10.1016/j.envsoft.2021.105205.

Cea, L., Costabile, P., 2022. Flood risk in urban areas: modelling, management and adaptation to climate change. A review. Hydrology 9 (3), 50. https://doi.org/10.3390/hydrology9030050.

Chen, T., Sun, J., Zhang, Z., Xiao, Z., Zheng, L., Chai, H., Lin, B., 2025. High-performance computing in urban flood modeling: a study on spatial partitioning techniques and parallel performance. J. Hydrol. 649, 132474. https://doi.org/10.1016/j.jhydrol.2024.132474.

Choi, H., Woo, H., Kim, M., Ryu, H., Lee, J.-H., Lee, S., Noh, S.J., 2025. FLO-SR: deep learning-based urban flood super-resolution model. J. Hydrol. 661, 133529. https://doi.org/10.1016/j.jhydrol.2025.133529.

Dong, B., Huang, B., Tan, C., Lin, K., Xia, J., Wang, X., Hu, Y., 2025a. City-scale high-resolution flood nowcasting based on high-performance hydrodynamic modelling. Int. J. Disaster Risk Reduct. 125, 105584. https://doi.org/10.1016/j.ijdrr.2025.105584.

Dong, B., Huang, B., Tan, C., Xia, J., Lin, K., Gao, S., Hu, Y., 2025b. Multi-GPU parallelization of shallow water modelling on unstructured meshes. J. Hydrol. 657, 133105. https://doi.org/10.1016/j.jhydrol.2025.133105.

Fewtrell, T.J., Duncan, A., Sampson, C.C., Neal, J.C., Bates, P.D., 2011. Benchmarking urban flood models of varying complexity and scale using high resolution terrestrial LiDAR data. Phys. Chem. Earth, Parts A/B/C 36 (7), 281–291. https://doi.org/10.1016/j.pce.2010.12.011.

Hou, J., Kang, Y., Hu, C., Tong, Y., Pan, B., Xia, J., 2020. A GPU-based numerical model coupling hydrodynamical and morphological processes. Int. J. Sediment Res. 35 (4), 386–394. https://doi.org/10.1016/j.ijsrc.2020.02.005.

Huang, M., Huang, B., Gu, L., Allen Huang, H.-L., Goldberg, M.D., 2015. Parallel GPU architecture framework for the WRF Single Moment 6-class microphysics scheme. Comput. Geosci. 83, 17–26. https://doi.org/10.1016/j.cageo.2015.06.014.

Kim, J.Y., Kang, J.-S., Joh, M., 2021. GPU acceleration of MPAS microphysics WSM6 using OpenACC directives: performance and verification. Comput. Geosci. 146, 104627. https://doi.org/10.1016/j.cageo.2020.104627.

Lee, J.-H., Lee, S., Kim, B., Choi, H., Noh, S.J., 2025. Evaluating the effects of spatial resolution on 2D pluvial flood modeling in urban built environments. J. Flood Risk Manag. 18 (3), e70105. https://doi.org/10.1111/jfr3.70105.

Lee, S., Nakagawa, H., Kawaike, K., Zhang, H., 2016. Urban inundation simulation considering road network and building configurations. J. Flood Risk Manag. 9 (3), 224–233. https://doi.org/10.1111/jfr3.12165.

Mielikainen, J., Huang, B., Wang, J., Allen Huang, H.-L., Goldberg, M.D., 2013. Compute unified device architecture (CUDA)-based parallelization of WRF Kessler cloud microphysics scheme. Comput. Geosci. 52, 292–299. https://doi.org/10.1016/j.cageo.2012.10.006.

Ming, X., Liang, Q., Jiang, J., 2025. Large-scale high-resolution hydrodynamic modelling of urban floods: some practical considerations. J. Hydro-environ. Res. 59, 100655. https://doi.org/10.1016/j.jher.2025.100655.

Morales-Hernández, M., Sharif, M.B., Kalyanapu, A., Ghafoor, S.K., Dullo, T.T., Gangrade, S., Kao, S.-C., Norman, M.R., Evans, K.J., 2021. TRITON: a Multi-GPU open source 2D hydrodynamic flood model. Environ. Model. Software 141, 105034. https://doi.org/10.1016/j.envsoft.2021.105034.

Noh, S.J., Lee, J.-H., Lee, S., Kawaike, K., Seo, D.-J., 2018. Hyper-resolution 1D-2D urban flood modelling using LiDAR data and hybrid parallelization. Environ. Model. Software 103, 131–145. https://doi.org/10.1016/j.envsoft.2018.02.008.

NVIDIA, 2025a. Multinode NVLink user guide - MNNVL user guide. https://docs.nvidia.com/multi-node-nvlink-systems/mnnvl-user-guide/. (Accessed 9 October 2025).

NVIDIA, 2025b. NVIDIA GB200 NVL72. https://www.nvidia.com/en-us/data-center/gb200-nvl72/. (Accessed 9 October 2025).

NVIDIA, 2025c. OpenACC Getting Started Guide. https://docs.nvidia.com/hpc-sdk/archive/25.7/. (Accessed 9 October 2025).

OpenACC, 2023. OpenACC Programming and Best Practices Guide. https://openacc-best-practices-guide.readthedocs.io/en/latest. (Accessed 9 October 2025).

Pickard, B.R., Daniel, J., Mehaffey, M., Jackson, L.E., Neale, A., 2015. EnviroAtlas: A new geospatial tool to foster ecosystem services science and resource management. Ecos. Serv. 14, 45–55. https://doi.org/10.1016/j.ecoser.2015.04.005.

Porter, A.R., Heimbach, P., 2025. Unlocking the power of parallel computing: GPU technologies for ocean forecasting. State Planet 5, 1–6. https://doi.org/10.5194/sp-5-opsr-23-2025.

Rautenbach, C., Trenham, C., Benn, D., Hoeke, R., Bosserelle, C., 2022. Computing efficiency of XBeach hydro- and wave dynamics on Graphics Processing Units (GPUs). Environ. Model. Software 157, 105532. https://doi.org/10.1016/j.envsoft.2022.105532.

Saleem, A.H., Norman, M.R., 2024. Accelerated numerical modeling of shallow water flows with MPI, OpenACC, and GPUs. Environ. Model. Software 180, 106141. https://doi.org/10.1016/j.envsoft.2024.106141.

Schubert, J.E., Sanders, B.F., 2012. Building treatments for urban flood inundation models and implications for predictive skill and modeling efficiency. Adv. Water Resour. 41, 49–64. https://doi.org/10.1016/j.advwatres.2012.02.012.

Wang, J., Hou, J., Li, S., Lu, B., Yan, Y., Mu, Y., 2025. Innovative flood prediction modeling: enhancing efficiency for large-scale simulations and analyzing responses to diverse impacting factors. J. Hydrol. 660, 133155. https://doi.org/10.1016/j.jhydrol.2025.133155.

Wang, P., Jiang, J., Lin, P., Ding, M., Wei, J., Zhang, F., Zhao, L., Li, Y., Yu, Z., Zheng, W., et al., 2021. The GPU version of LASG/IAP Climate System Ocean Model version 3 (LICOM3) under the heterogeneous-compute interface for portability (HIP) framework and its large-scale application. Geosci. Model Dev. 14 (5), 2781–2799. https://doi.org/10.5194/gmd-14-2781-2021.

Xie, J., Feng, X., Gao, T., Dong, C., Yin, B., Wu, C., 2025. Accelerating an implicit ocean model using CUDA C. Appl. Ocean Res. 163, 104740. https://doi.org/10.1016/j.apor.2025.104740.

Xu, A., Li, B.-T., 2023. Multi-GPU thermal lattice Boltzmann simulations using OpenACC and MPI. Int. J. Heat Mass Tran. 201, 123649. https://doi.org/10.1016/j.ijheatmasstransfer.2022.123649.

Xu, A., Li, B.-T., 2024. Particle-resolved thermal lattice Boltzmann simulation using OpenACC on multi-GPUs. Int. J. Heat Mass Tran. 218, 124758. https://doi.org/10.1016/j.ijheatmasstransfer.2023.124758.

Xu, S., Huang, X., Oey, L.-Y., Xu, F., Fu, H., Zhang, Y., Yang, G., 2015. POM.gpu-v1.0: a GPU-based Princeton Ocean model. Geosci. Model Dev. 8 (9), 2815–2827. https://doi.org/10.5194/gmd-8-2815-2015.

Xue, W., Wang, H., Roy, C.J., 2024. CPU–GPU heterogeneous code acceleration of a finite volume Computational Fluid Dynamics solver. Future Gener. Comput. Syst. 158, 367–377. https://doi.org/10.1016/j.future.2024.04.049.

Yang, L., Hou, J., Wang, X., Wang, P., Wang, Y., 2025. Application of the 2D high-resolution eco-hydraulics model based on GPU acceleration technology in the Upper Yellow River. Environ. Model. Software 188, 106393. https://doi.org/10.1016/j.envsoft.2025.106393.

Zhang, Y.-Y., Xu, W.-J., Tian, F.-Q., Du, X.-H., 2025. A Multi-GPUs based SWE algorithm and its application in the simulation of flood routing. Adv. Water Resour. 201, 104985. https://doi.org/10.1016/j.advwatres.2025.104985.